International Journal of Research and Reviews in Applied Sciences
ISSN: 2076-734X, EISSN: 2076-7366
Volume 1, Issue 1(October 2009)

REVIEW OF AGILE METHODOLOGIES IN SOFTWARE DEVELOPMENT

[1]MALIK HNEIF, [2]SIEW HOCK OW

[1]Department of Software Engineering, University of Malaya, Kuala Lumpur, Malaysia-50603

[2]Assoc. Prof., Department of Software Engineering, University of Malaya, Kuala Lumpur, Malaysia-50603

ABSTRACT

During the past forty years, new software development approaches were introduced to fit the new cultures of the software development companies. Most software companies nowadays aim to produce valuable software in short time period with minimal costs, and within unstable, changing environments. Agile Methodologies were thus introduced to meet the new requirements of the software development companies. This paper presents a review of three agile approaches including Extreme Programming, Agile Modeling, and SCRUM, describes the differences between them and recommends when to use them.

Keywords: *Software development approaches; agile methodologies; extreme programming; agile modeling; SCRUM.*

1. INTRODUCTION

As we notice, software development is expanding. Software has merged into many diverse fields, and is becoming more complex. Changing requirements from customers is making it even more difficult. Old software development approaches are not able to satisfy the new requirements of the market in the best way, anymore. As a result, new software development approaches are evolved, as agile methodologies, mainly to solve such problem. The new methodologies include modifications to software development processes, to make them more productive and flexible.

This paper has the following structure: Section 2 briefs the history of development approaches. Section 3 explains the evolvement of software development towards agile methodologies, and presents the values and concepts of agile development. It also covers the main and most used agile methodologies. Section 4 describes the limitations to apply agile methodologies, and the last section concludes the paper.

2. HISTORY OF SOFTWARE DEVELOPMENT APPROACHES

Programming started with structured languages as Fortran in 1954 [4], and then evolved to object-oriented languages in the 1960s [15]. Similarly, software development approaches have evolved during time.

A software development approach guides the developer through the software development process. To develop software, the developers usually choose a software development approach, which usually divides the development process into phases, and for each phase, the developer has to apply the guidelines that the selected approach provides for that phase.

The first development approach came into existence after the software crisis in the 1970s [13]. Software engineers tried to reduce the effect of the crisis and stop it by following structured methodologies for software development. These methodologies divide the software development process into phases, so that the developer focuses his/her efforts on one phase at a time. Using these methods, the number of failed or uncompleted software projects was reduced, the cost and development time of software projects were decreased, and the effect of the software crisis is limited [13].

After the structured methodologies, software development approaches evolved into Object-Oriented (OO) methodologies, which include: Unified Process and Rational Unified Process (RUP). They include not only the objects and object oriented principles, but also the best practices from structured methodologies [13].

3. AGILE METHODOLOGIES

At the early years of software development, most of the users' requirements were fairly stable, and development followed the plans without major changes. However, as software development involved more critical and dynamic industrial projects, new difficulties emerged according to the growth of companies. These difficulties include [1][19]:

- Evolving requirements: customer requirements are changing due to evolving business needs or legislative issues. Most of the customers do not have a clear vision about the specifications of their requirements at the early stages. Some customers realize what their true requirements are only when they use an application that does not really meet their needs. Another source of change comes from experiences gained during the development.
- Customer involvement: lack of customer involvement leads to higher chances of project failure. Many companies usually do not allocate any effort for customer involvement.
- Deadlines and budgets: often, customers do not accept failure. On the other hand, companies usually offer low budgets, tight deadlines, while at the same time, requiring high demands, and all of this is because of competition in the markets.
- Miscommunications: one cause of the misunderstanding of requirements is the miscommunication between developers and customers. For example, each party uses its own jargon, and this leads to misunderstanding of customer's needs.

With the existence of such problems, the OO software development methodologies cannot satisfy the objectives of software development companies. New development methodologies have to be applied in order to overcome these problems.

A number of IT professionals started to work individually on new approaches to develop software. The results of their researches were a set of new development methodologies that have many common features. When they met in 2001 in conference in Utah [1], they created the so called: Agile Manifesto. These approaches were developed based on the same rule that the best way to verify a system is to deliver working versions to the customer, then update it according to their notes. Agile authors built their methodologies on four principles. First, the main objective is to develop software that satisfies the customers, through continuous delivering of working software, and getting feedback from customers about it. The second principle is accepting changes in requirements at any development stage, so that customers would feel more comfortable with the development process. The third principle is the cooperation between the developers and the customers (business people) on a daily basis throughout the project development. The last principle is developing on a test-driven basis; that is to write test prior to writing code. A test suite is run on the application after any code change [1].

Agility in short means to strip away as much of the heaviness, commonly associated with traditional software development methodologies, as possible, in order to promote quick response to changing environments, changes in user requirements, accelerate project deadlines, and the like [7]. Agile methodologies prefer software development over documentation. Their philosophy is to deliver many working versions of the software in short iterations, then update the software according to customers' feedback. Applying this philosophy will help to overcome the problems mentioned earlier, by welcoming changes, satisfying user requirements, faster development, and at the end, users will have just the system they need.

Agile methodologies include:

- Extreme Programming
- Agile Modeling

- SCRUM
- Crystal methodologies family
- Feature-Driven Development
- Adaptive Software Development

3.1 Extreme Programming (XP)

Extreme Programming was introduced by Kent Beck in 2000. Being an emerging agile methodology, XP offers a number of practices, values and principles which are advised to be adopted in order to run a software development project [3]. XP is a package of several practices and ideas, most of which are not new. The combination and packaging of all of these is, however, new [11]. Extreme Programming was in fact targeted especially at small co-located teams developing non-critical products. It has been suggested that the early adopters of agile methods have been small high-tech product companies [16]. Currently, however, it has already been proven at many companies of all different sizes and industries worldwide [8].

XP provides a list of simple, specific, and seemingly naïve principles and values that guide the software development process throughout the main four phases of software development: planning, coding, designing, and testing (Figure 1). The main purpose is to deliver what the customer needs, at the time it is needed. In addition to this, one of the main reasons of its success is its ability to accept changes at anytime during the development. XP also emphasises teamwork; experiences from all stakeholders are employed to meet the specific goals, and within the given constraints [8].

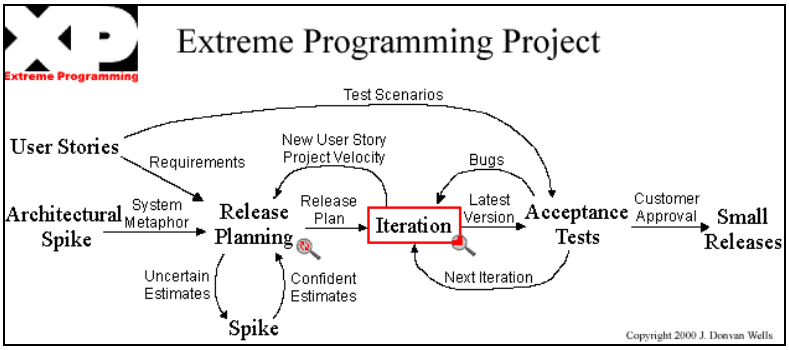

Figure 1. The flow in a project using XP [8]

This agile methodology improves software development in four ways. Firstly, high communications between the developers and their customers. Second, the simplicity of design. Third, the continuous modification according customer's feedback, by delivering a large number of working software to the customer. Fourth, encouraging the customer and developers to get to the correct requirements by the develop-feedback cycle [8]. Those four items are called XP values.

There are some programming principles that are encouraged by XP. One of them is that, simplicity and flexibility will reduce maintenance costs of the software in the future. Another programming practice is the intensive and robust testing mechanism, which will reduce the number of bugs reported later by the customer after delivering the final version of the application. Embracing changes is also encouraged by XP, since customers are most likely to notice new opportunities for improving the system to meet their goals, while still in the development process. Moreover, XP encourages creating high quality code [8].

Many other rules and practices are provided by XP, for all the phases of software development. These include: creating user stories; making small frequent releases; dividing the project into iterations; creating spike solutions to reduce risk; availability of the customer; writing code to agreed standards; pair

programming; leaving optimization until last; all the code is unit tested; creating a test case for each bug found; just to name a few [8][11].

However, XP is not best suited for any project. There are some conditions that help to decide whether to apply the XP methodology for a software development project or not. Some projects have unclear or dynamic requirements, in such case XP will succeed, while other methodologies will fail. For projects with high risk that appear to be a new challenge for the developing company, XP practices can help to lower this risk, and increase the possibility of success. On the other hand, XP is not best suited for companies with large teams; it works best for teams with 2 to 12 members. Additionally, it assumes strong cooperation and communication between the developers and the customers. If this is not the case (not possible for any reason such as distance), then XP will not give good results. Another condition is testability. Applying XP requires intensive testing for the software, from the first day until delivery. However, if the nature of the software to be developed does now allow this, XP will not be the best to apply. Mainly, the core purpose of XP is to deliver the software that is needed, when it is needed [8].

One empirical study conducted by Kuppuswami et al. (2003) found that increasing effort (independent variable) into XP core practices reduced the total effort (dependent variable) needed to create the system [7]. Furthermore, XP is being experimented in different ways to make it fit to the specific needs of the projects as well as the development teams [18]. In summary, XP is the coding of what the customer specifies, and testing that code, to ensure that the prior steps in the development process have accomplished what the developers intended to do [7].

3.2 Agile Modeling (AM)

Modeling is an important step in software development. It enables software developers to think about complex issues before addressing them in programming. Agile Modeling (AM) was established by Scott Ambler in 2002. It is a collection of values, principles, and practices for modeling software that can be applied on a software development project in an effective and light-weight manner [2]. Agile Modeling was built to be adapted to, and used with existing methodologies, as XP and RUP, aiming to allow a developer to build a software system that truly meets the customer's needs.

The values of AM, which are considered to be an extension to the values of XP include: communication, simplicity, feedback, courage, and humility. Humility means to admit that you may not know everything; others may know things that you do not know, and thus, they may provide useful contribution to the project [2].

Again, the principles of AM are quite similar to those of XP, such as assuming simplicity, embracing changes, incremental change of the system, and rapid feedback. In addition to these principles, AM principles include the knowledge of the purpose for modeling; having multiple effective models; the content is more important than the representation; keeping open and honest communication between parties involved in the development process; and finally, to focus on the quality of the work [2].

The practices of AM have some commonalities with those of XP, too. An agile modeler needs to follow these practices to create a successful model for the system. AM practices highlight on active stakeholder participation; focus on group work to create the suitable models; apply the appropriate artifact as UML diagrams; verify the correctness of the model, implement it and show the resulting interface to the user; model in small increments; create several models in parallel; apply modeling standards; and other practices [2].

Agile Model Driven Development (AMDD) is the agile version of model driven development. To apply AMDD, an overall high level model for the whole system is created at the early stage of the project. During the development iterations, the modeling is performed as planned per iteration. Usually, AM is applied along with other methodologies, such as Test Driven Development (TDD), and Extreme Programming (XP), to get the best results [2].

AM basically creates a mediator between rigid methodologies and lightweight methodologies, by suggesting that developers communicate architectures through applying its practices to the modeling

process [7]. In a nut, agile modeling defines a collection of values, principles, and practices which describe how to streamline the modeling and documentation efforts. It is usually applied in conjunction with agile implementation techniques for good results.

3.3 SCRUM

SCRUM methodology was initiated by Ken Swaber in 1995. It was practiced before the announcement of Agile Manifesto. Later, it was included into agile methodology since it has the same underlying concepts and rules of agile development. SCRUM has been used with the objective of simplifying project control through simple processes, easy to update documentation and higher team iteration over exhaustive documentation [6].

SCRUM shares the basic concepts and practices with the other agile methodologies, but it comprises project management as part of its practices. These practices guide the development team to find out the tasks at each development iteration.

In addition to the practices defined for agility, one main mechanism recommended by SCRUM is to build a backlog. A backlog is a place where one can see all requirements pending for a project, sized based on complexity, days or some other unit of measure the team decides. Inside a product backlog, there is a simple sentence for each requirement; something that will be used by the team to start discussions and putting details of what is needed to be implemented by the team for that requirement [6].

For the team of SCRUM, three main roles are defined as shown in Fig 2. The first role is the product owner, who mainly would be the voice of business. The second role is the SCRUM team which comprises developers, testers, and other roles. This team would make initial contact with customer and identify the need for a new product. SCRUM master, the third role, is responsible for keeping the team focused on the specific goals, and help the team members to solve problems when they appear [6][17].

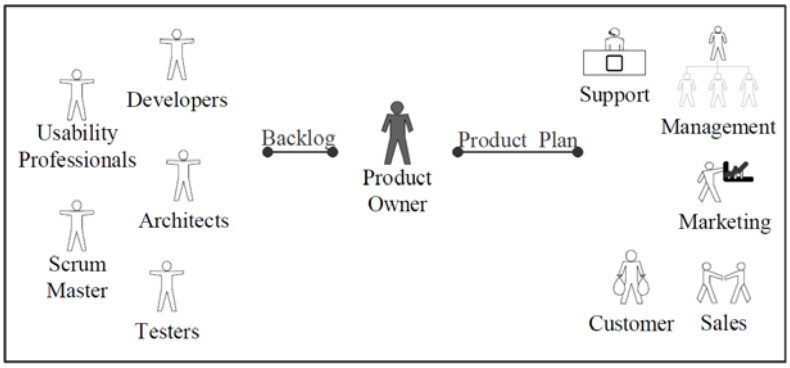

Figure 2. Key roles and interaction artifacts in SCRUM [17]

The process of development using SCRUM divides the project into phases. In each phase, one feature is fully developed, tested, and become ready to go to production. The team does not move to a new phase until the current phase is completed. Whether what is being done adds value to the process or not, is the main concern of each phase.

Current studies on traditional SCRUM development have shown that despites its advantages, it is not best suited for products where the focus is on usability [17]. It fails to address usability needs of the user, because product owners keep their focus mainly on business issues and forget about usability. Since product owners usually come from business background, they lack the experience, skills, and motivation to design for user experiences. Moreover, traditional agile methodologies are not concerned about the user experience vision, which drives the architecture and is essential for ensuring a coherent set of user experiences [10]. According to Mona Singh, U-SCRUM, is an agile methodology for promoting usability [17].

Briefly, SCRUM is considered an iterative, incremental methodology of software development. It was proposed for software development projects, and at the same time, it can be used as a program management approach.

3.4 Other agile methodologies

In this paper, three main agile methodologies that have been widely used in software development are discussed. Besides these three methodologies, there are some other software development methodologies lay under the agile umbrella. They include Crystal methodologies, Feature-Driven Development, and Adaptive Software Development.

Crystal Methodologies were established by Alistar Cockburn in 2000. They concentrate on efficiency and habitability as components of project safety. Each of the Crystal methodologies requires certain roles, policy standards, products and tools to be adopted [12]. Crystal Clear, which is one of the Crystal methodologies, can be applied to development teams of six to eight members, working on non-life critical systems. It focuses on people, not processes of artifacts [5].

Feature Driven Development (FDD) was founded by Jeff De Luca and Peter Coad. It combines some practices recognized in the industry into one methodology. These practices are all determined from a client-valued functionality (feature) viewpoint. As of other agile methodologies, its key goal is to deliver tangible, working software repeatedly in a timely manner [9].

Adaptive Software Development (ASD) was created in 2000 by Jim Highsmith. It has grown out of the Rapid Application Development (RAD). Like other agile methodologies, ASD aims to increase a software organization's responsiveness while decreasing development overhead [14]. It embodies the belief that continuous adaptation of the process to the work at hand is the normal state of affairs.

4. LIMITATIONS OF AGILE METHODOLOGIES

Agile development aims to support early and quick development of working code that meets the needs of the customer. Agile supporters claim that code is the only deliverable that matters, whereas, agile opponents found that emphasis on code will lead to memory loss, because the amount of documentation and modeling done is not enough [19].

There are some limitations to apply agile methodologies [19]. First one is that agile methodologies are not suitable for green-field engineering and not suitable for maintenance, since there will be not much documentation for the systems. The second limitation is that agile methodologies depend heavily on the user involvement, and thus, the success of the project will depend on the cooperation and communication of the user. Another limitation is that agile methodologies concentrate work quality on the skills and behaviours of the developers, as the design of the modules and sub-modules are created mainly by single developer. When developing software to be reusable, then agile methodologies will not provide the best way. This is because they focus on building systems that solve specific problems, and not the general ones. Agile methodologies work best for teams with relatively small number of members, and hence, they will not work well for teams with large number of members.

To get the advantages of applying agile methodologies in the development, there is a set of assumptions that are assumed to be true. To mention some are: cooperation and face to face relation between the customers and the development team; evolving and changing requirements of the project; developers having good individual skills and experiences; in addition to many more [19]. If these assumptions do not apply to a software development project, then it is better to look for other methodologies to apply for the development process, in order to get better results.

5. CONCLUSION

Software development methodologies have evolved since the 1970s. Agile methodologies came into existence after the need for a light way to do software development in order to accommodate changing requirements environment. Agile methodologies provide some practices that facilitate communication between the developer and the customer, and under go develop-deliver-feedback cycles, to have more specific view of the requirements, and be ready for any change at any time. The main aim of agile methodologies is to deliver what is needed when it is needed.

Agile methodologies include a set of software development approaches. They have some variations, but still they share the same basic concepts. The main agile methodologies that are being used include XP, Agile Modeling, and SCRUM. XP is the coding of what the customer specifies, and the testing of that code. Agile Modeling defines a collection of values, principles, and practices which describe how to streamline modeling and documentation efforts. SCRM, on the other hand, supports management role in software development.

Agile methodologies are not best suited for all projects. When communication between the developer and the customer is difficult, or when the development team includes mainly beginners, agile methodologies will not give the best results. These methodologies exhibit optimum results when there is a strong communication between the developer and the customer, and the development team compromises skilled team members. When there is a big chance for misunderstanding the exact customer's requirements, or when the deadlines and budgets are tight, then Agile methodologies are among the best software development approaches to apply.

6. REFRENCES:

[1] Agile Alliance. Manifesto for Agile Software Development. [Online] Retrieved 16th March 2009. Available at: http://www.agilemanifesto.org .

[2] Agile Modeling Home Page. Effective Practices for Modeling and Documentation. [Online] Retrieved 17th March 2009. Available at: www.agilemodeling.com .

[3] K. Beck, and C. Andres, Extreme Programming Explained: Embrace Change (2nd Edition), Addison-Wesley, Boston, 2004.

[4] Computer Programming. [Online] Retrieved 5th April 2009. Available at: http://en.wikipedia.org/wiki/Programming

[5] Crystal Clear (software development). [Online] Retrieved 22nd March 2009. Available at: http://en.wikipedia.org/wiki/Crystal_Clear_(software_development)

[6] M. Cristal, D. Wildt and R. Prikladnicki, Usage of SCRUM Practices within a Global Company. Global Software Engineering, 2008. ICGSE 2008. IEEE International Conference on, 2008, 222-226.

[7] J. Erickson, K. Lyytinen and K. Siau, Agile Modeling, Agile Software Development, and Extreme Programming: The State of Research. In Journal of Database Management, 16(4), 2005, 88-100.

[8] Extreme Programming. What is Extreme Programming? [Online] Retrieved 18th March 2009. Available at: www.extremeprogramming.org

[9] Feature Driven Development. [Online] Retrieved 18th March 2009. Available at: http://en.wikipedia.org/wiki/Feature_Driven_Development

[10] J. Ferreira, J. Noble, and R. Biddle., Up-Front Interaction Design in Agile Development. In Agile Processes in Software Engineering and Extreme Programming, Springer , Berlin / Heidelberg , 2007, 9-16.

[11] D. Karlström, Introducing Extreme Programming - An Experience Report. In proceedings 3rd International Conference on eXtreme Programming and Agile Processes in Software Engineering, XP 2002, Sardinia, Italy.

[12] F. Keenan, Agile process tailoring and problem analysis (APTLY). In Software Engineering, 2004. ICSE (2004). Proceedings. 26th International Conference on, 45- 47.

[13] C. Klimeš and J. Procházka, New Approaches in Software Development. In Acta Electrotechnica et Informatica, 6(2), 2006.

[14] F. Maurer and S. Martel, Extreme programming. Rapid development for Web-based applications. In Internet Computing, IEEE, 6(1), 2002, 86-90.

[15] Object-oriented programming. [Online] Retrieved 20th March 2009. Available at: http://en.wikipedia.org/wiki/Object_oriented

[16] C. Schwaber and R. Fichera, Corporate IT leads the second wave of agile adoption. Forrester Research, Inc, 2005.

[17] M. Singh, U-SCRUM: An Agile Methodology for Promoting Usability. In Ag. AGILE '08. Conference, Toronto, 2008, 555-560.

[18] B. Tessem, Experiences in Learning XP Practices: A Qualitative Study. In Extreme Programming and Agile Processes in Software Engineering. 2003, 131-137.

[19] D. Turk, R. France and B. Rumpe, Limitations of agile software processes. In Proceedings of the Third International Conference on Extreme Programming and Flexible Processes in Software Engineering, 2002.